CS 585 Assignment 2

Don Johnson

January 29, 2014

1 Assignment 2 (100 points)

1.1 Learning Objectives

1.2 Technical Task (30 points)

Source code for technical task in ZIP file: Assignment2.cpp, Assignment2.hpp, Gui.cpp and Operators.cpp. All output images are in a subfolder in the ZIP file called "assignment2/Data."

1.3 Programming Assignment Questions (28 points)

1. Run your program on the image yellowEgg.jpg both by thresholding the gradient magnitude and by using the Canny edge detector. Choose a threshold and smoothing level that leads to a nice contour and filled region of the egg. Results below:

   Note: Smoothing value units are one standard deviation and threshold values are pixel integer values. Unlike the gradient edge detector, the Canny has both a lower and upper threshold that I set to be threes times the lower threshold. Consequently, the Canny output files have two thresholds in their name.

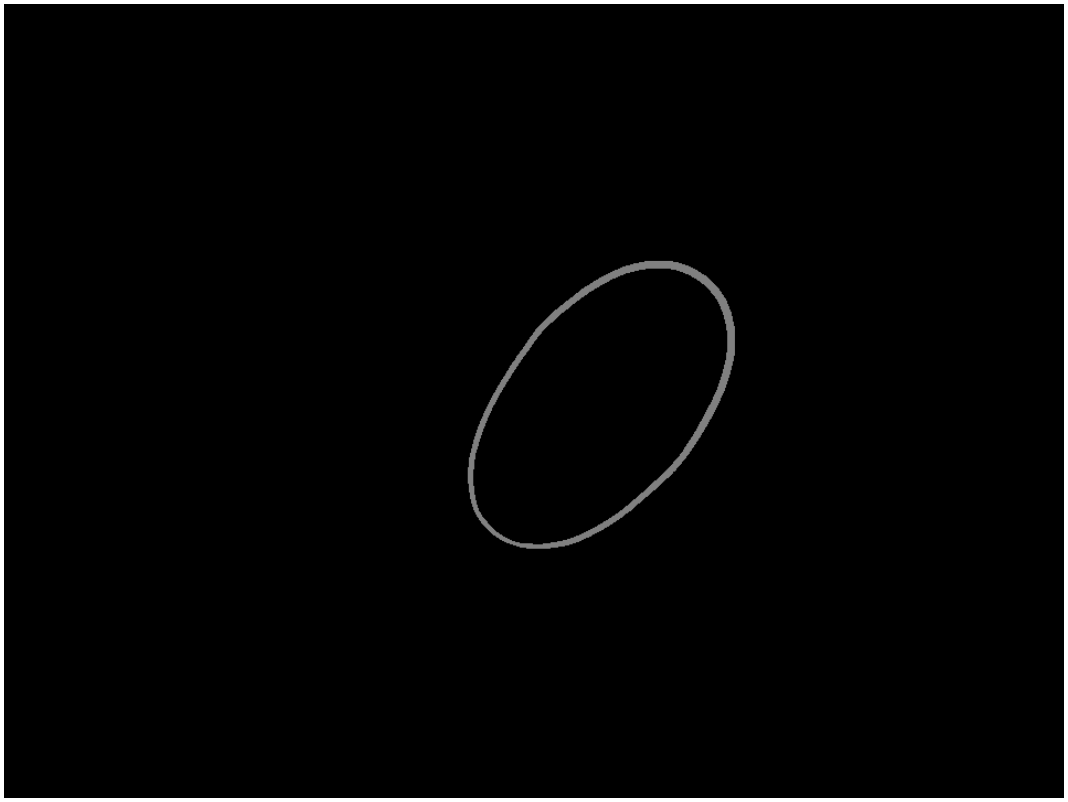

Figure 1: SoloYellowEgg-Gradient-edges-2pt5-82.png

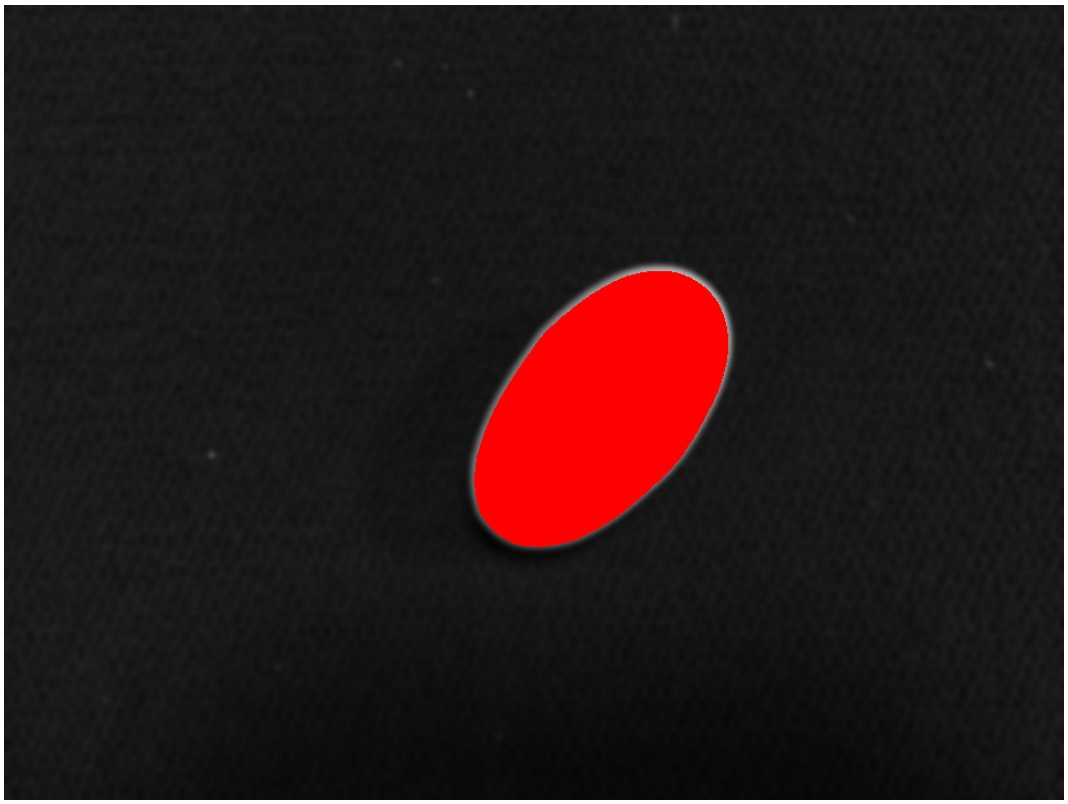

Figure 2: SoloYellowEgg-Gradient-result-2pt5-82.png

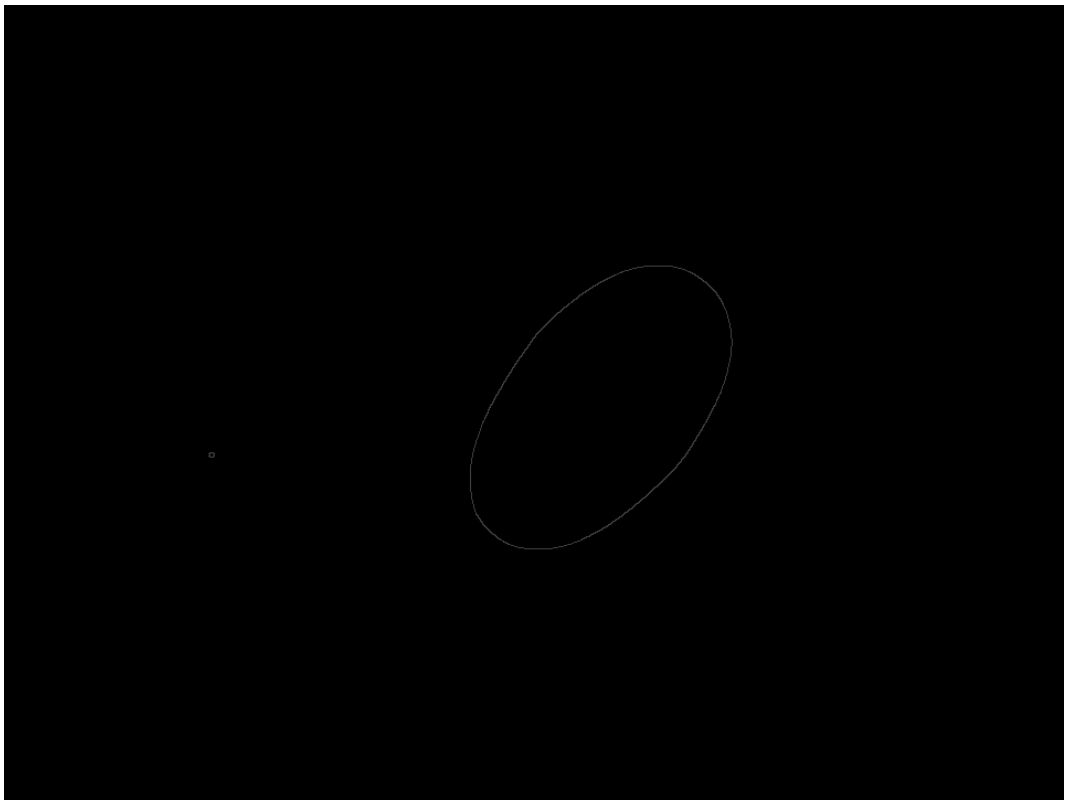

Figure 3: SoloYellowEgg-Canny-edges-1-129-387.png

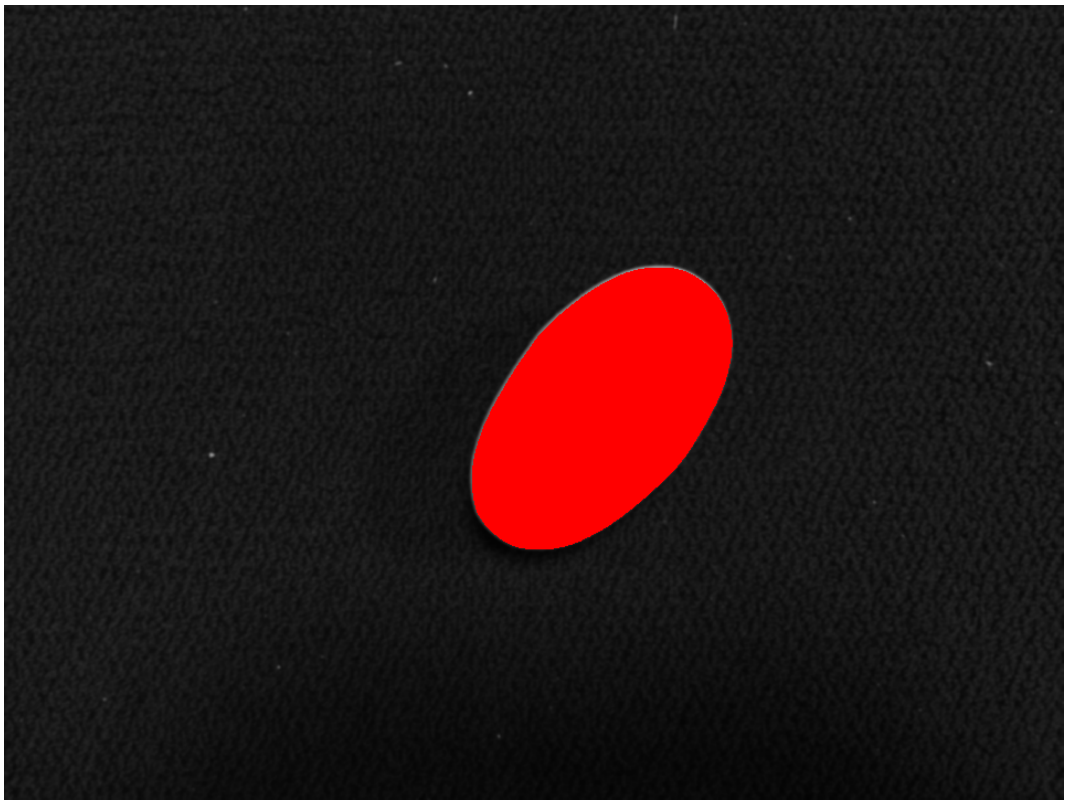

Figure 4: SoloYellowEgg-Canny-result-1-129-387.png

2. Run your program on the image "eggsAndCube.jpg." Choose a smoothing level that you feel gives good contours in the gradient magnitude, and then for each of the red, green, blue, and yellow eggs, choose the highest threshold that gives a clean result. Results below:

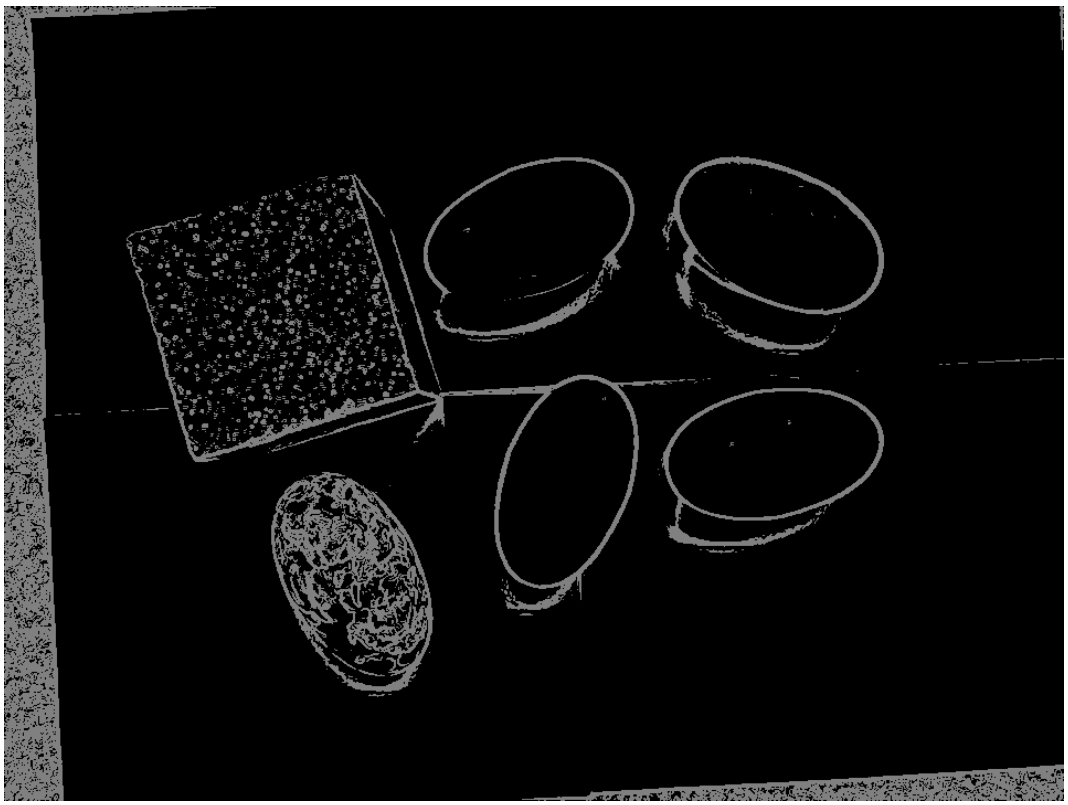

Figure 5: redEgg-edges-0pt5-41.png

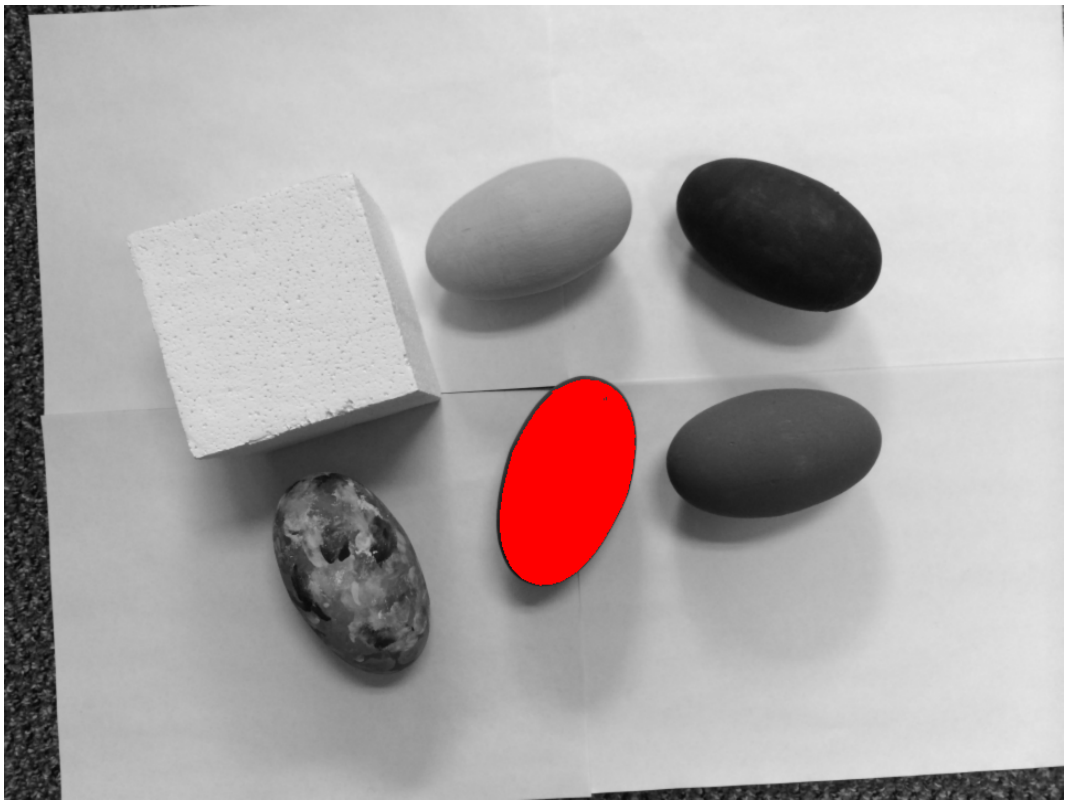

Figure 6: redEgg-result-0pt5-41.png

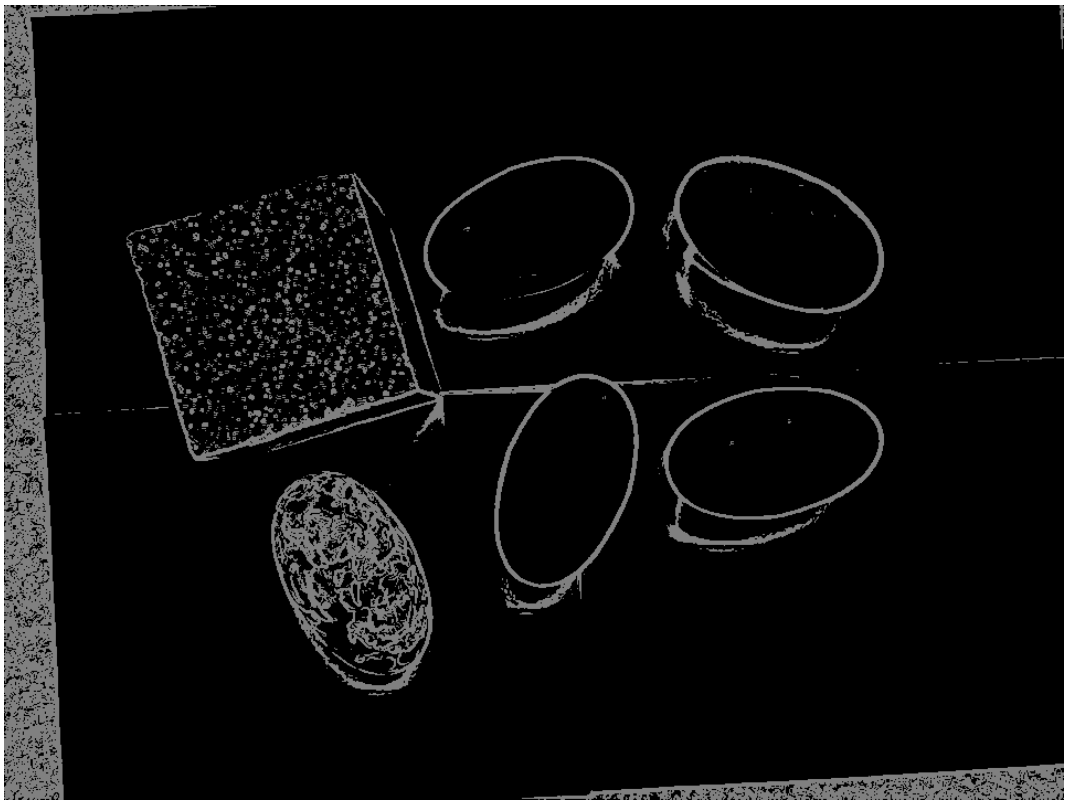

Figure 7: greenEgg-edges-0pt5-41.png

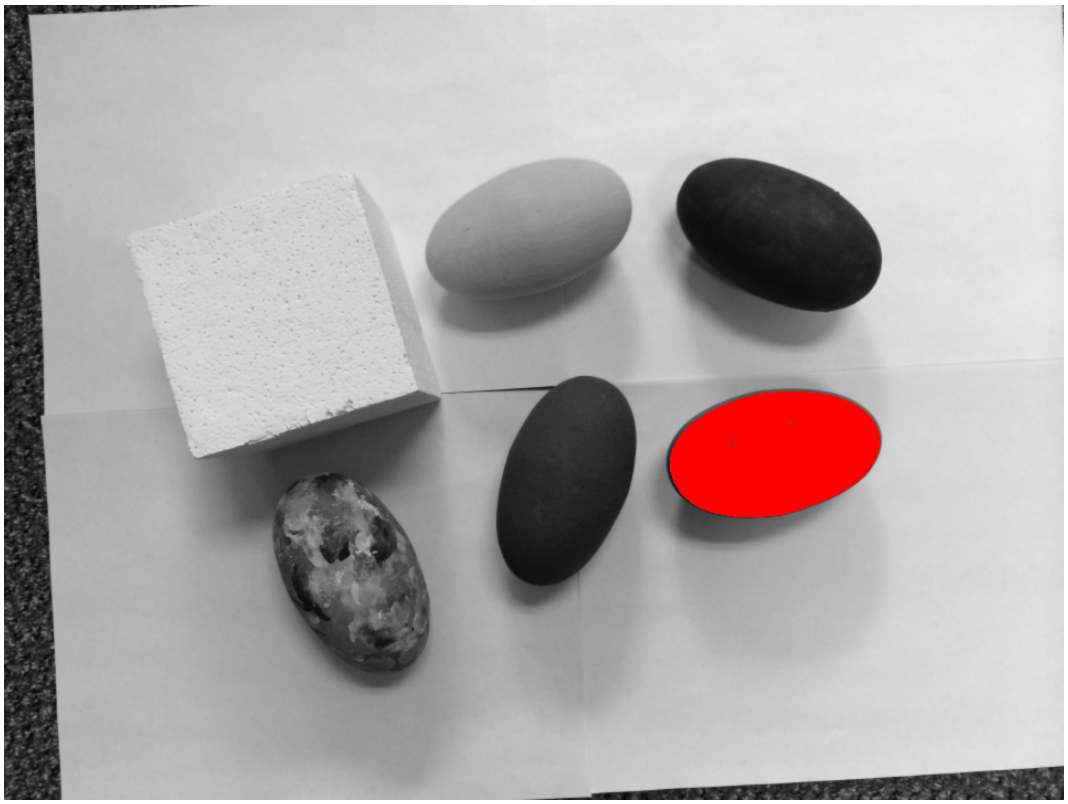

Figure 8: greenEgg-result-0pt5-41.png

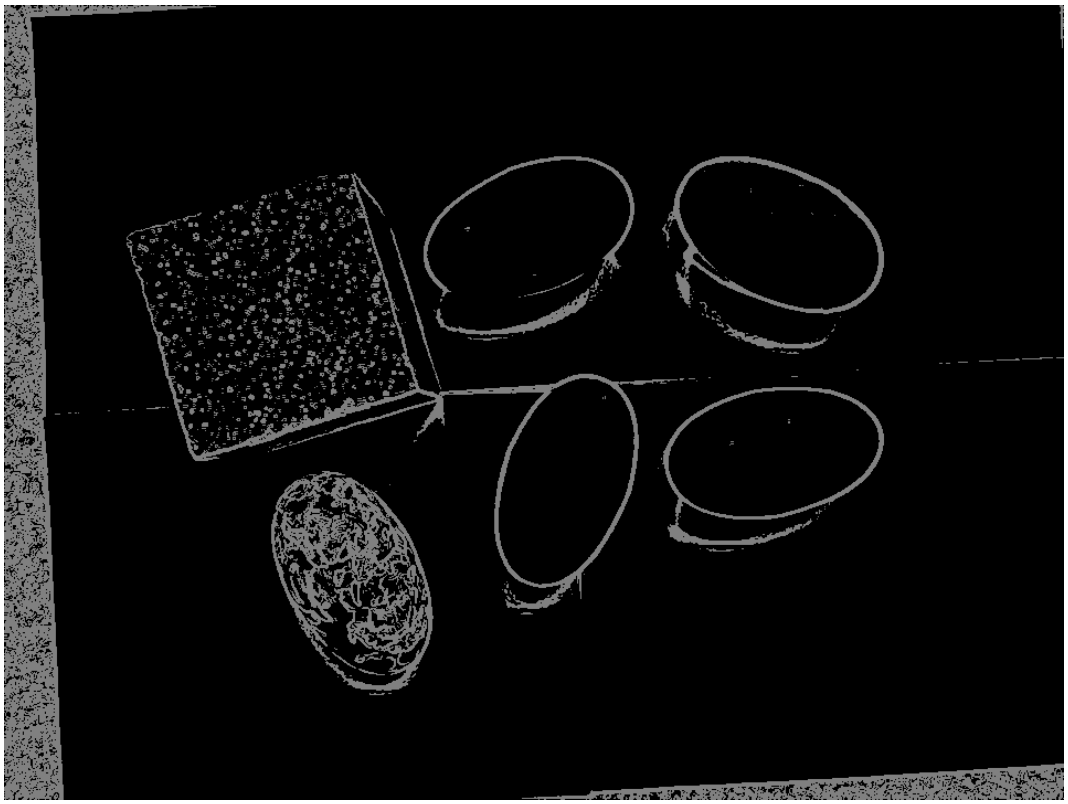

Figure 9: blueEgg-edges-0pt5-42.png

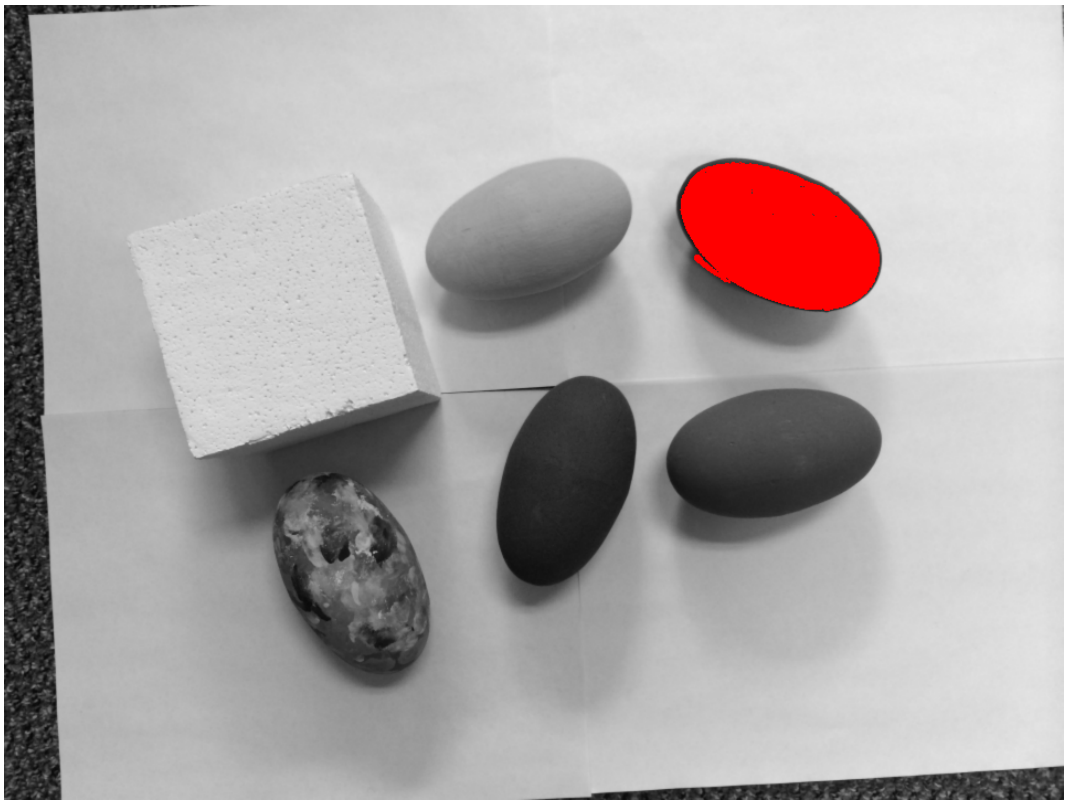

Figure 10: blueEgg-result-0pt5-42.png

3. Are you able to find any parameters that will allow you to identify the cube (or a face of the cube)?

   Harder with "gradient," because of the texture of the cube. Need low threshold and higher smoothing, however with "Canny" with dilation it is easy to select the same face on the cube cleanly (Smoothing sigma = 1.5, Canny thresholds = 20 and 60, Dilation kernel size = 3). See the results below (right vertical face):

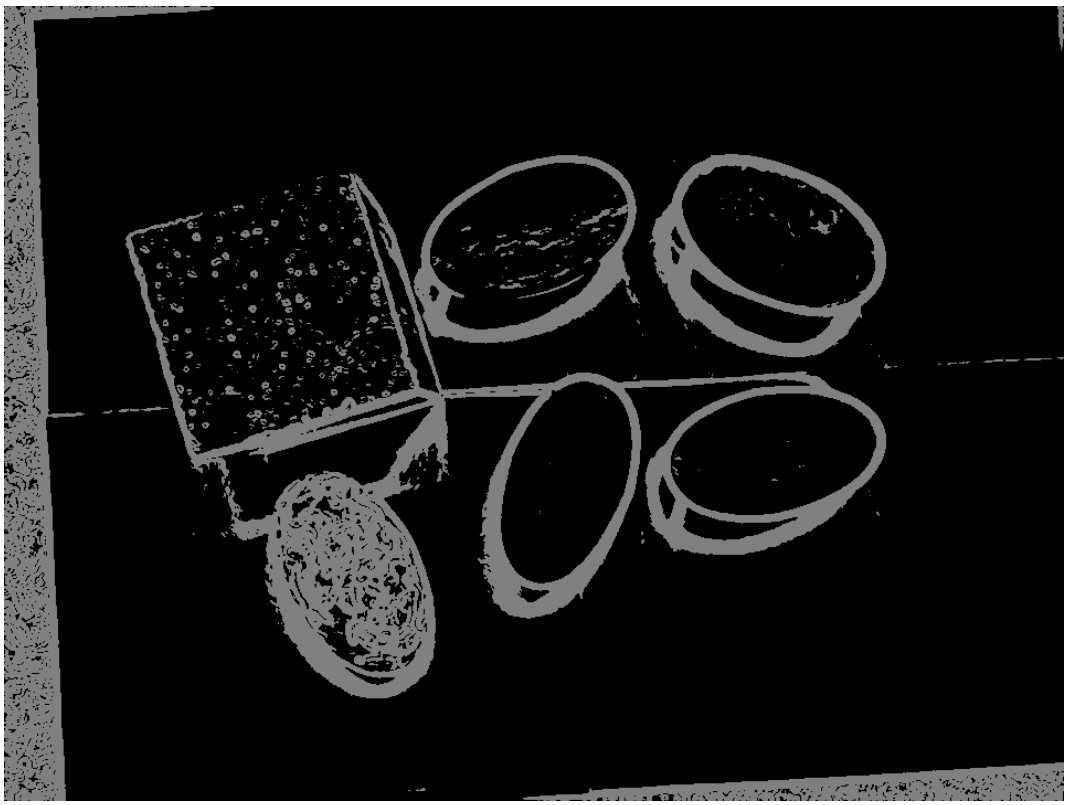

Figure 11: cube-edges-1pt5-16.png

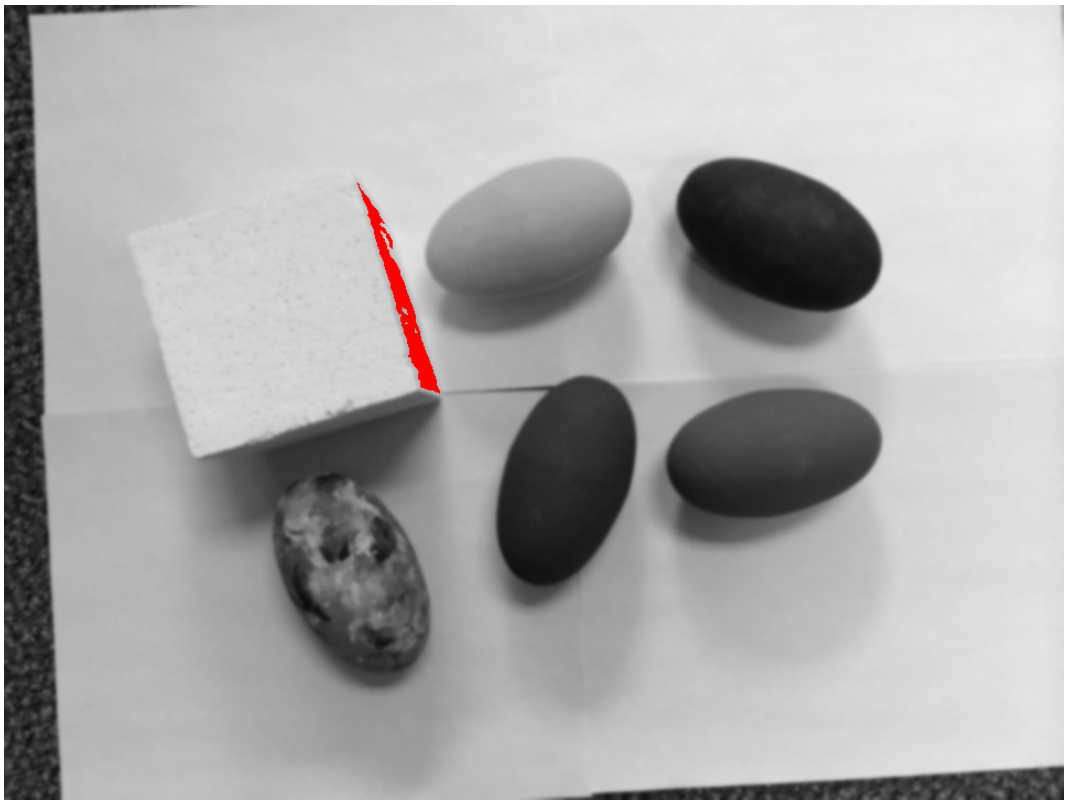

Figure 12: cube-result-1pt5-16.png

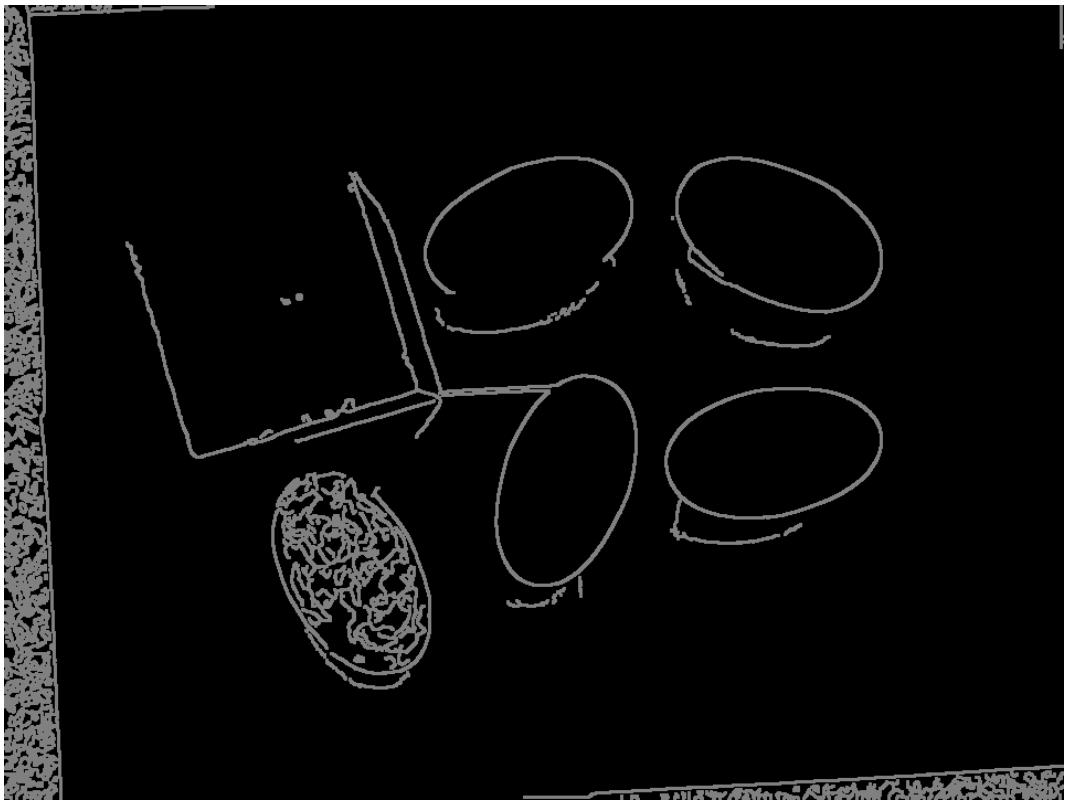

Figure 13: cube-canny-dilation-edges.png

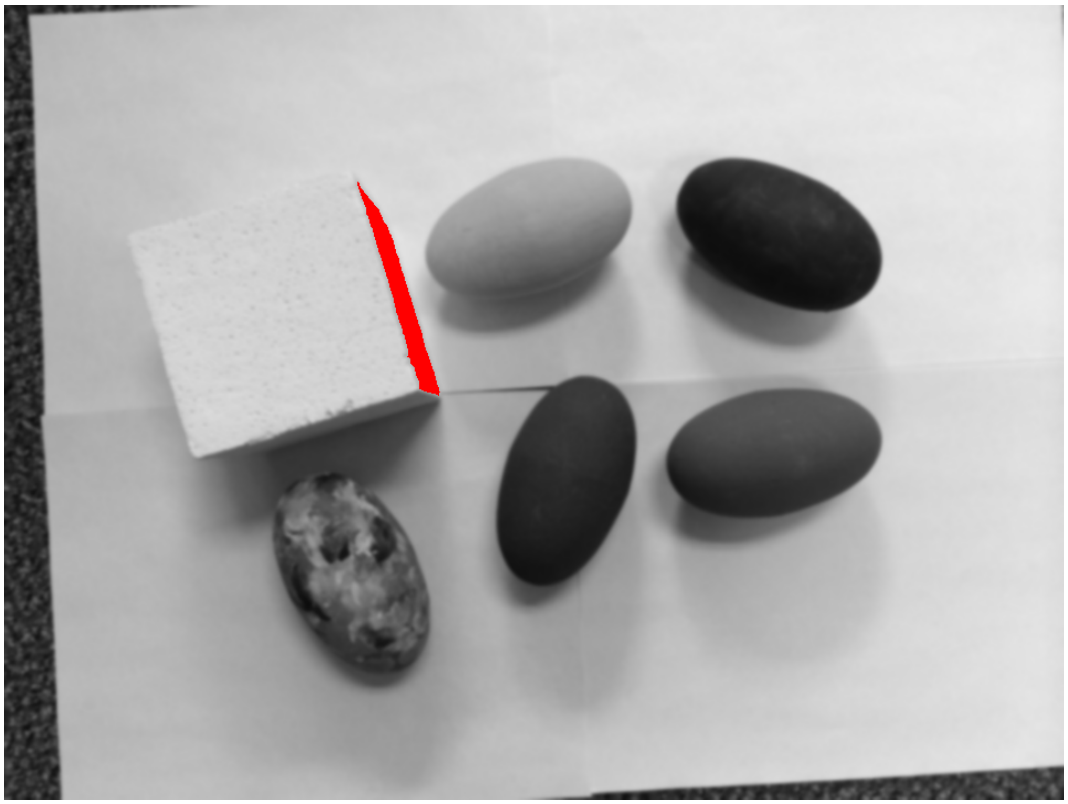

Figure 14: cube-canny-dilation-result.png

4. Use your program on the image "camoEggsOnPaper.jpg." If you knew that you were looking for textured objects on a plain background, what steps could you combine in order to highlight an entire egg, even though there are lots of internal edges? What type of binary image analysis steps could you perform in order to segment the entire egg?

   Canny with high thresholds (low=149 and high=447) and 3x3 kernel dilation can the closest to working. Nothing I tried to close off the eggs edges without almost filling the eggs with mask. Initial smoothing wasn't needed and didn't help for Canny, because Canny does a Gaussian smoothing as part of its procedure. Results below:

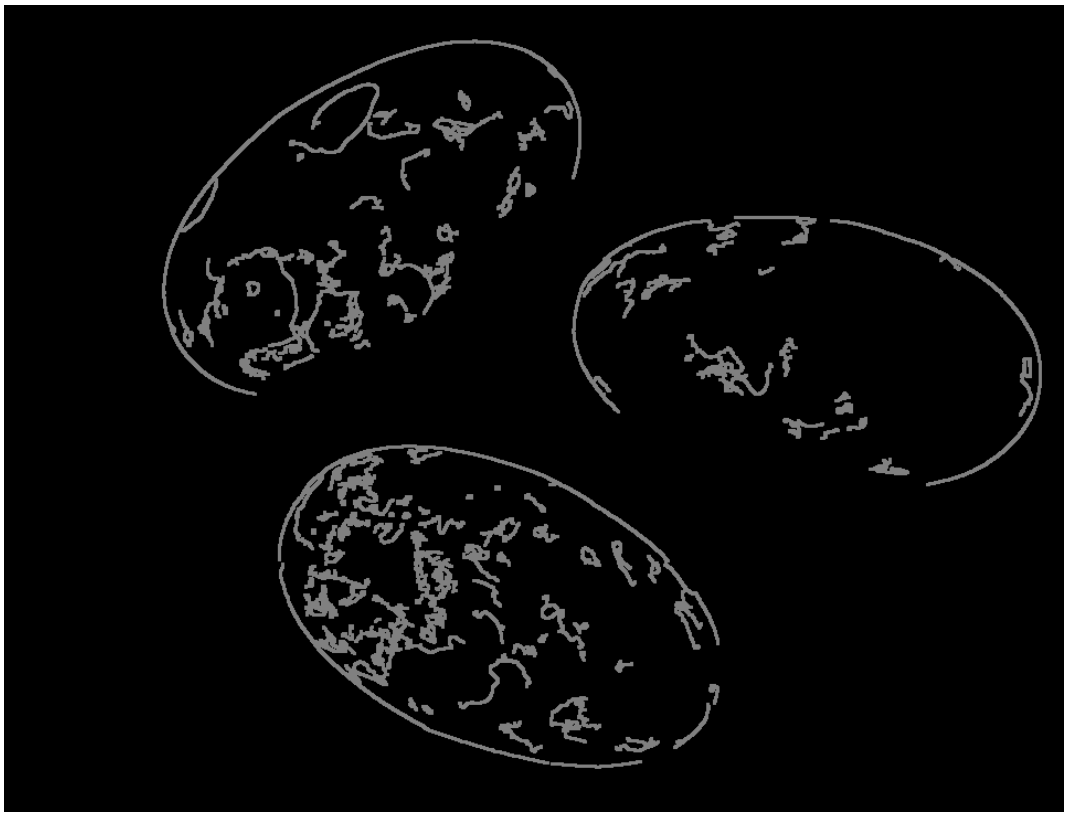

Figure 15: camoEggsOnPaper-canny-dilation-edges.png

5. Run your program on the image rainbowSpheres.jpg. You may find that some areas of the image are easier to segment than others. Try clicking on a few image regions (spots or stripes on the balls). Results below:

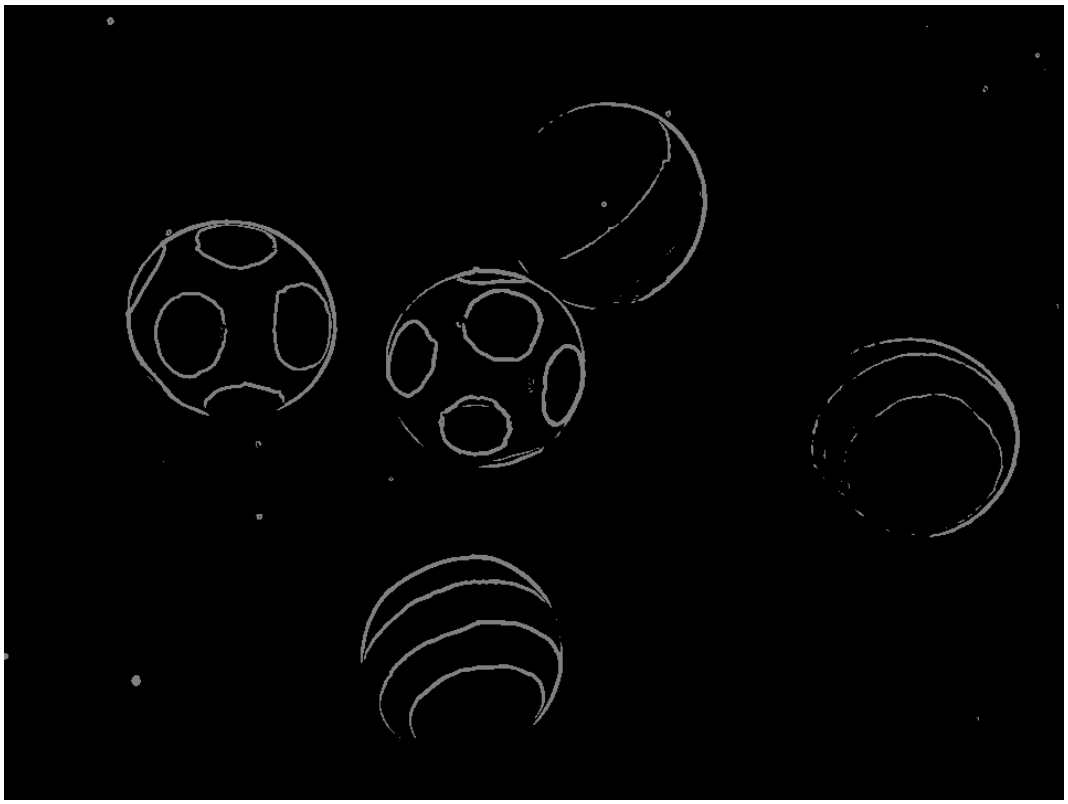

Figure 16: rainbowEasy-edges-1-94.png

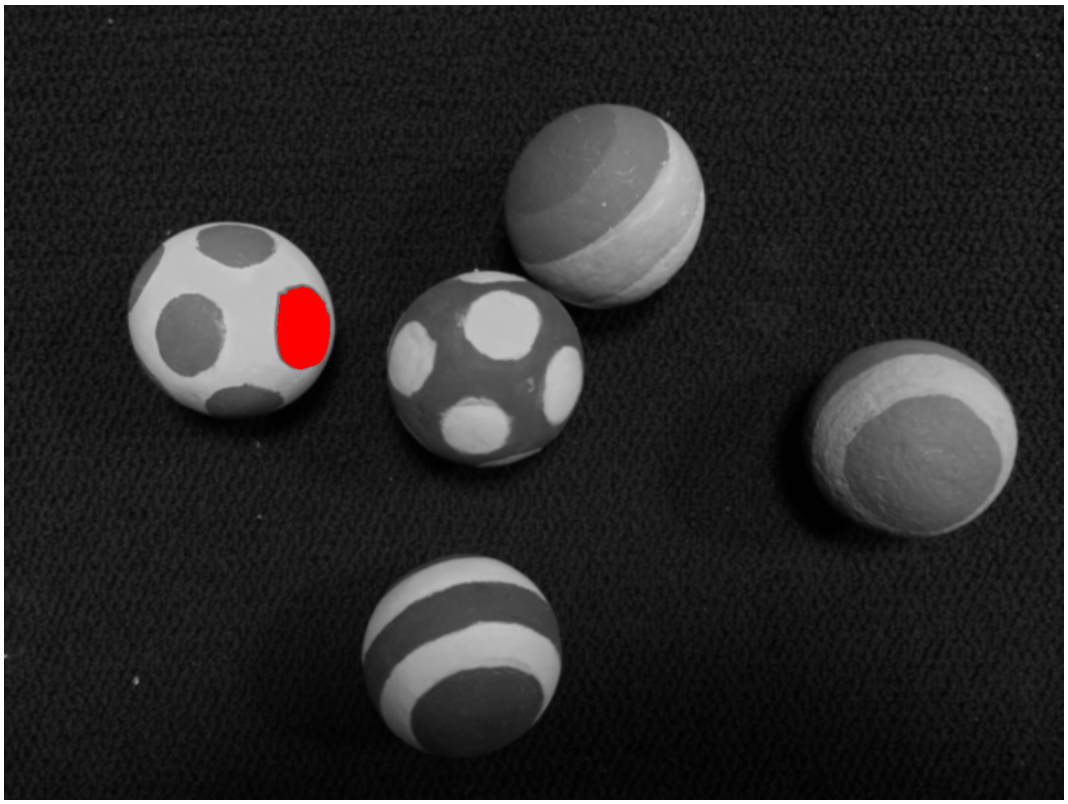

Figure 17: rainbowEasy-result-1-94.png

6. Identify an area of the image that you think should be reasonably easy to pick out, but that the program is having trouble with. (For example, the purple and blue stripes on top of the rainbow ball). What is happening that causes the entire background to be filled when you attempt to fill the region bounded by edges starting in one of these areas?

The entire background is filled because the strip I picked out was missing part of its left edge.

7. Use morphological operators to manipulate your edge map to help close gaps where the region-filling is leaking out so that you can fill in difficult region. Create before and after images of your troublesome regions.

   I used 1 sigma smoothing, a lower threshold and a 3x3 kernel dilation to fill the gap. Results below:

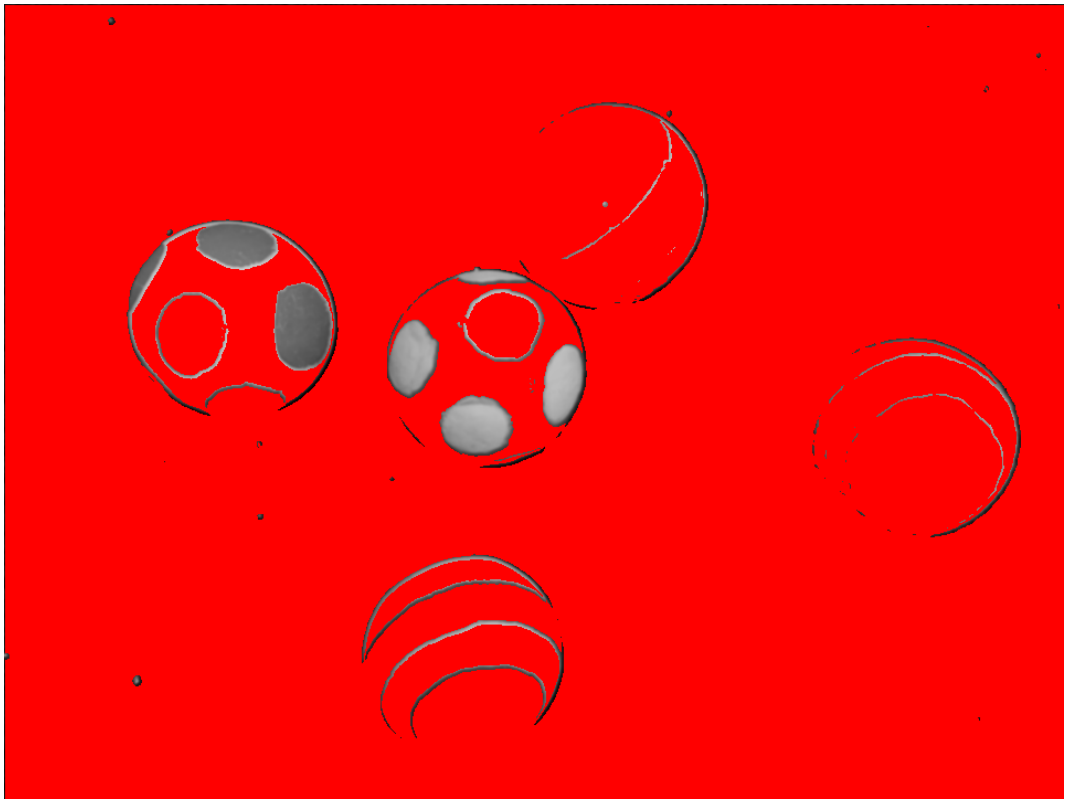

Figure 18: rainbowHard-before-result-1-94.png

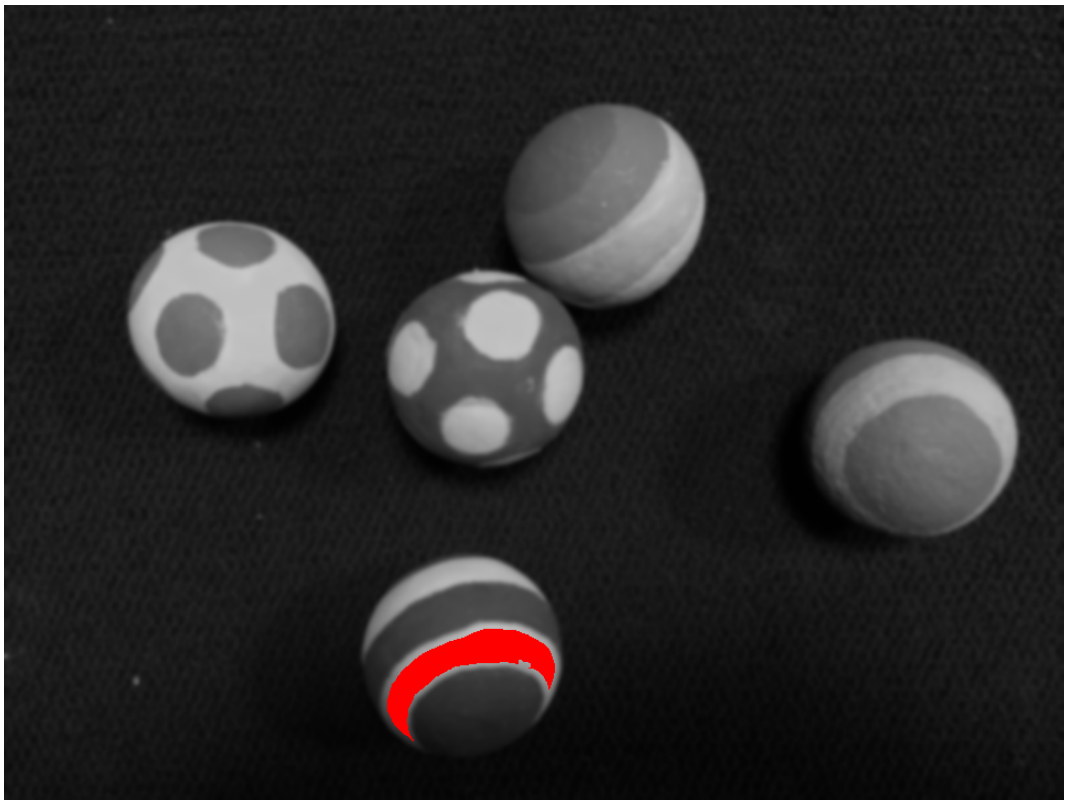

Figure 19: rainbowHard-after-result-1-33.png
; ;

1.4 Lab Follow-up Questions (39 points)

1. Under the folder for Lab2, there is a folder called "gradients" containing several image where I have prepared several images using the gradient formula. Remember that the gradient of a function is just a vector containing the partial derivatives of the function with respect to each variable. You take a partial derivative of a function using the normal rules of calculus, but you treat all variables other than the one you are interested in as constants.

   (a) Write an expression for the gradient of I, given the formula that I used to generate the images.
   Holding y, θ constant and taking the derivative with respect to x yields:

   $$\frac{\partial I}{\partial x} I(x,y) = \cos\theta$$

   Holding x, θ constant and taking the derivative with respect to y yields:

   $$\frac{\partial I}{\partial y} I(x,y) = \sin\theta$$

   (b) For each of three values calculate what you anticipate the values of the gradient will be:

   $$\begin{aligned} \nabla I(\theta) &= (\sin\theta, \cos\theta) \\ \nabla I(0^\circ) &= (0,1) \\ \nabla I(45^\circ) &= \left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right) \\ \nabla I(90^\circ) &= (1,0) \end{aligned}$$

   (c) Using the program from the Lab, inspect the gradient of the images named "gradient_theta=0.png", "gradient_theta=45.png", "gradient_theta=90.png." Did you find any discrepancies? Yes, the 0 and ninety degree values were opposite of what I expected because the dI/dx and dI/dy values were reversed. These are the values I got from running the program against the three images:

Degree	dX	dY
0	1	0
45	0.7	0.7
90	0	1

(d) Using the program from the Lab, inspect the gradient of the image named "gradient_mysteryTheta.png." Report the values that you find and use them to determine the angle I used to generate the image. Non-noisy mystery theta Arctan(dI/dy,dI/dx) = (255/191) = 53 degrees

(e) Histogram is gaussian before smoothing of mystery gradient noisy, using mean instead of three point gives Arctan(dI/dy,dI/dx) = = (147.33/167.03) = 41.41 degrees

(f) Histogram is just a few spikes after smoothing did seem to help much. Noisy mystery theta Arctan(dI/dy,dI/dx) = [(64+128+191)/(159+191+223)] = (383/573) = 56 degrees

(g) The image "gradient sigmoid.png" has been generated with such a function (given below). By inspecting the gradient of the image at the center point, estimate the angle that I used to generate the image. The gradient looks like a a negative 45 degrees.

2. Write down the kernel associated witth logic. It is a 9x9 kernel since there are nine terms:

1	2	1
2	4	2
1	2	1

3. I replicated the nearest neighbor row and then column to complete the convolution and populate the boundary pixels:

Problem 3. Convolution

	1	2	3	4	5	6	7
1	17	17	50	17	20	46	46
2	17	17	50	17	20	46	46
3	69	69	34	37	57	-13	-13
4	29	29	50	26	24	46	46
5	27	27	34	25	35	13	13
6	54	54	33	38	46	38	38
7	54	54	33	38	46	38	38

4. These are the erosion and dilation of the original image:

Problem 4. Erosion

	1	2	3	4	5	6	7						1	2	3	4	5	6	7
1	1	1	0	0	0	0	0						1	0	0	0	0	0	0	0
2	0	1	0	1	0	0	0		0	1	0		2	0	0	0	0	0	0	0
3	1	0	1	0	0	1	0		1	1	1	===>	3	0	0	0	0	0	0	0
4	0	1	0	0	0	0	0		0	1	0		4	0	0	0	0	0	0	0
5	0	1	1	0	0	0	0						5	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0						6	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0						7	0	0	0	0	0	0	0

Problem 4. Dilation

	1	2	3	4	5	6	7						1	2	3	4	5	6	7
1	1	1	0	0	0	0	0						1	1	1	1	1	1	1	0
2	0	1	0	1	0	0	0		0	1	0		2	1	1	1	1	1	1	1
3	1	0	1	0	0	1	0		1	1	1	===>	3	1	1	1	1	1	1	1
4	0	1	0	0	0	0	0		0	1	0		4	1	1	1	1	0	1	1
5	0	1	1	0	0	0	0						5	1	1	1	1	1	1	0
6	0	0	0	0	0	0	0						6	1	1	1	1	0	0	0
7	0	0	0	0	0	0	0						7	0	1	1	0	0	0	0

2 Lecture preparation (3 points)

2.1 Preparation for Lecture 4

Read Viola-Jones and Summed Area Table articles.

2.2 Preparation for Lecture 5 (3 pts)

In Lecture 5, we will discuss image pyramids for performing multi-scale feature detection. Under the folder for Assignment 2, there is a folder called "camoEggsPyramid." Use your program from section 1.2 to perform edge detection on each of the images in the folder. What do you notice about the way the edge-finding and region-filling works on each of the images in the pyramid?

The most obvious thing is that the images get much smaller and more pixelated as you move up the levels. Also, it seams easier to detect complete edges as you move up the levels.